Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Upon entrance of this amendment, claims 1, 3-4, 9-11, 14-22, 27, 29-30, 35-37, 40-48, 53-56, 58, 60-61, 66-68, and 70-79 will be pending.

By this amendment, independent claims 1, 27, 53 & 58 are amended to further characterize the dynamically altering as being done "upon receipt of the request waiting on the response." Support for this amendment can be found throughout the application, and in particular, at lines 7-11 of the Abstract. Additionally, the subject matter of dependent claims 2 & 28 (canceled herein) is incorporated into the respective independent claims 1 & 27. Independent claims 53 and 58 are similarly amended as well. Thus, the amendments presented herewith are not believed to comprise new matter.

In the final Office Action, claims 1-4, 9-11, 14-22, 27-30, 35-37, 40-48, 53-56, 58-61, 66-68 & 71-79 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Schoening et al. (U.S. Patent No. 6,205,465; hereinafter, "Schoening") in view of Furlani et al. (U.S. Patent No. 5,995,998; hereinafter, "Furlani"). This rejection is respectfully, but most strenuously, traversed to any extent deemed applicable to the independent claims presented herewith.

An "obviousness" determination requires an evaluation of whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art. In evaluating claimed subject matter as a whole, the Federal Circuit has expressly mandated that functional claim language be considered in evaluating a claim relative to the prior art. Applicants respectfully submit that the application of these standards to the independent claims presented herewith leads to the conclusion that the recited subject matter would not have been obvious to one of ordinary skill in the art based on the applied patents.

Applicants' recite a technique for managing thread pools of a computing environment (e.g., claim 1). This technique includes receiving from a first requester of the computing environment a request to be processed. This request is waiting on a response from a second requester of said computing environment, and the response is to be serviced by a thread pool selected from a set of one or more eligible thread pools. The technique further includes, upon receipt of the request waiting on the response, dynamically altering the set of one or more eligible thread pools to provide an altered thread pool set of one or more eligible thread pools, wherein a thread pool of the altered thread pool set is to service said response, and wherein the dynamically altering comprises setting a pool mask associated with the response to indicate one or more eligible thread pools. Applicants respectfully submit that at least the feature of dynamically altering the set of one or more eligible thread pools, as well as the timing of the dynamically altering in applicants' claimed invention (i.e., upon receipt of the request waiting on the response), and the manner in which the dynamically altering is performed (i.e., by setting a pool mask associated with the response to indicate the one or more eligible thread pools), are not taught, suggested or implied by Schoening or Furlani, alone or in combination.

Schoening describes a parallel processing technique for a multi-threaded environment in which threads are organized by receipt of execution components that have a partial order determined by preconditions and resource requirements. Schoening also discloses determining a final order of execution of these components, some of which may run in parallel on separate threads (see Abstract and Col. 4, lines 1-29 thereof). This processing technique is quite different from the present invention, which recites, in part, dynamically altering the set of one or more eligible thread pools to provide an altered thread pool set of the one or more eligible thread pools, wherein a thread pool of the altered thread pool set is to service the response from the second requester (e.g., claim 1). This dynamically altering provides, for example, an increase in the number of thread pools in a set from one pool to two pools. Such an increase in the number of thread pools can, for example, provide a second thread pool to service the response which is different from the one thread pool that was available (prior to the dynamically altering) to service both the response and the request that was waiting on the response. This servicing of the response on, for example, a second thread pool in the altered thread pool set avoids a deadlock situation that could have occurred prior to the dynamically altering if no thread in the thread pool was available to service the response required to complete the processing of the request waiting on the response. The technique in Schoening does not alter a set of thread pools to, for example, change the number of thread pools in a set from one to two. Instead, Schoening applies preconditions to assemble threads from execution components (see Title). Assembling threads in Schoening is directed to ordering the dispatch of components on threads, where some are run in parallel, without regard to whether the threads are selected from a set of thread pools that has been altered (see Abstract; Col. 23, lines 14-21).

Not only does Schoening fail to disclose dynamically altering a set of one or more eligible thread pools, applicants respectfully submit that it also lacks a teaching, suggestion or implication that the dynamically altering occurs upon receipt of the request waiting on the response, as recited by the claims presented herewith. The altering recited in applicants' invention is dynamic because it takes place during processing of a request (i.e., upon receipt of the request waiting on the response; see, e.g., claim 1). It is through such dynamic alteration of a set of eligible thread pools that the present invention avoids a type of deadlock that occurs when requests being processed require a response to be processed (e.g., to access a locked data file that the response will unlock), but also leave no threads available for the response to be processed. In Schoening, the determination of execution order is not performed during processing of a request (i.e., not upon receipt of a request). Instead, this determination in Schoening is performed during the startup of the ANI interface, which is prior to processing of any requests (see Col. 22, line 67 – Col. 23, line 3; Col. 39, lines 48-51; Col. 41, lines 25-35; FIG. 6A).

An additional deficiency in Schoening as applied to the present invention is its failure to disclose the manner by which applicants' recited dynamically altering occurs. That is, Schoening lacks a teaching, suggestion or implication of dynamically altering a set of one or more eligible thread pools, wherein the dynamically altering includes setting a pool mask associated with the response to indicate the one or more eligible thread pools, as recited by the claims presented herewith (e.g., claim 1). As is well-known in the art, a mask can include a pattern of bits whose "on" and "off" states indicate two different states. A pool mask, for example, can be set to indicate the thread pools eligible to be used to dispatch requests or responses thereon (see specification, page 14, lines 13-14). If a thread pool is eligible for processing, a bit in the pool mask is set to its "on" state. If, however, the thread pool is not eligible for processing, the bit is set to "off" (specification, page 15, lines 1-4). A careful reading of Schoening reveals no discussion of setting a pool mask to indicate eligible thread pools.

In the final Office Action, it is stated that the determination of whether the component can be run in parallel reads on applicants' recited "waiting on a response" while the determination is made (final Office Action, page 5, paragraph 16). Applicants respectfully submit that a careful reading of Schoening reveals no express teaching of parallel execution determination that is done while waiting on a response. Instead, Schoening teaches such determining without waiting on a response because, at the time of this determination, there are no threads dispatched yet that could process a response (Col. 22, line 64 - Col. 23, line 13; Col. 41, lines 25-35). Applicants also submit that to determine such parallel execution while waiting on a response (i.e., after processing of threads begins) would increase the complexity of the required coding from independent developers and negate one of the stated goals of the Schoening technique (i.e., to minimize the effects on code writing; see Col. 3, lines 19-22). As such, applicants submit that applying preconditions to determine parallel execution prior to the dispatch of any thread is a significant part of Schoening's technique and thus, in addition to lacking an express teaching, Schoening does not suggest or imply such determination while waiting on a response.

Paragraphs 17 & 19 of the final Office Action stated that Schoening teaches altering thread pools and that this altering is arguably dynamic because the whole process is dynamic. Applicants respectfully submit that the presence of a dynamic process in Schoening is not relevant to applicants' remarks. As stated above, there is a dynamic quality to Schoening's determination of execution order (i.e., determined during ANI startup), but this determination does not teach, suggest or imply dynamically altering a set of eligible thread pools. Further, the claims presented herewith include qualifications of applicants' recited dynamically altering, and thus, applicants request that the above-referenced final Office Action statements be reconsidered in view thereof.

Paragraph 18 of the final Office Action stated that the partial order of Schoening consists of eligible thread pools and that applicants' reference to "silence" was not understood. Applicants respectfully submit that whether Schoening's partial order teaches eligible thread pools is not relevant to applicants' remarks, which are directed to not merely eligible thread pools, but to applicants' recited <u>dynamically altering</u> the set of one or more eligible thread pools. Moreover, applicants' statement of "silence as to eligible thread pools alone . . ." (Response dated January 21, 2003, page 5, line 16) is a restatement of the discussion above (i.e., applicants' remarks are <u>not</u> directed to whether Schoening is or is not silent as to eligible thread pools alone).

The final Office Action also stated that Schoening teaches thread pool masking at Col. 41, lines 1-5 thereof (see final Office Action, page 3, paragraph 5). The cited section of Schoening describes a partial order as defining an order of execution of code blocks, and an evaluation sequence that is stored in a Partial Order object. Each member of the evaluation sequence identifies one of a plurality of code blocks (see FIG. 5a). As such, this section of Schoening describes multiple identifiers for ordering code blocks, which differs from a pool mask whose bits store one of two states associated with a thread pool (i.e., eligible/ineligible to server requests and responses). Thus, applicants respectfully submit that this section provides no teaching, suggestion or implication of the setting of a pool mask, as recited by the claims presented herewith.

To summarize, Schoening does not teach, suggest or imply dynamically altering a set of one or more eligible thread pools, let alone dynamically altering upon receipt of the request waiting on the response, or dynamically altering by setting a pool mask associated with the response to indicate the one or more eligible thread pools. Thus, since Schoening fails to describe, suggest or imply multiple features of applicants' claimed invention, applicants submit that it does not render applicants' invention obvious. Further, applicants respectfully submit that Furlani does not overcome the deficiencies of Schoening as applied to the present invention.

Furlani describes a technique for locking groups of interrelated objects in a multi-threaded computing environment to minimize lock contention. The technique also includes merging groups and group locks (see Abstract and col. 2, lines 50-65 thereof). For example, in Furlani, when formerly independent objects A and B with separate locks become related, the separate locks merge so that a single lock applies to both A and B. This is very different from the present invention, as recited in claim 1, wherein the change in the thread pools is not a merge, but a dynamic alteration of a set of one or more eligible thread pools into an altered thread pool set. Further, the groups in Furlani are quite different from the thread pools claimed by applicants. Furlani's groups are groups of objects (e.g., data structures), rather than pools of threads (i.e., independently executable parts of a program). Still further, applicants submit that Furlani does not describe or suggest dynamically altering upon receipt of a request waiting on a response, nor by the setting of a pool mask associated with the response to indicate the one or more eligible thread pools.

The final Office Action stated that "the groups and/or merging locks both alter processing pools dependent on whether they are locked or merged" (page 6, paragraph 20). Applicants respectfully submit that a careful reading of Furlani uncovers no teaching or suggestion that groups and/or merging locks alter a processing pool, let alone a set of one or more eligible thread pools, as recited by the present invention. Moreover, applicants respectfully submit that the final Office Action does not make it clear as to how a non-active entity, such as an object in Furlani, is suggestive of an active entity, such as thread in a thread pool in the present invention. In any event, applicants request reconsideration of the above-referenced statement in the final Office Action in light of the amended claims presented herewith.

Based on the foregoing, both Schoening and Furlani are believed to fail to teach, suggest or imply multiple features of applicants' claimed invention. Thus, applicants submit that the combination of Schoening and Furlani also fails to teach, suggest or imply multiple features of the present invention.

For all of the above reasons, applicants respectfully submit that independent claim 1, as well as the other independent claims presented are patentable over the combination of Schoening and Furlani. Additionally, the dependent claims are believed patentable for the same reasons as the independent claims from which they directly or ultimately depend, as well as for their own additional characterizations.

For example, claim 19 further recites that the thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response. Various types of deadlocks are well-known in the computer art and a person of ordinary skill in the art can identify a particular type given an appropriate context. Such a context is provided in applicants' specification and is illustrated by the following example: client A has a hold on Resource X (e.g., a lock on a file) and is currently updating Resource X. Client B then requests Resource X. The server breaks the lock by sending a callback to Client A. Client A responds to the callback before Client B is granted access to the resource. All threads in the thread pool may be already processing client requests that are waiting for the callback response from Client A. In this case, there are no threads in the thread pool to handle Client A's response, thereby causing a deadlock wherein Client A's response and the waiting requests are prevented from completing their respective processing (see specification, page 2, line 17 – page 3, line 4).

As recited by claim 19, deadlock is avoided when the thread pool of the altered thread pool set is to service the response, rather than the thread pool selected from the original set of one or more eligible thread pools (i.e., the set prior to the alteration). Thus, deadlock avoidance is an advantageous feature of dynamically altering the set of one or more eligible thread pools. In contrast, Schoening does not teach or suggest deadlock avoidance. Instead, Schoening is directed to a parallel processing technique that provides extensibility (i.e., supports new devices and new services without major revision of the network management system) and promotes efficiency (Col. 3, lines 32-49). Applicants note that Schoening mentions deadlock (Col. 3, lines 24-30; Col. 54, lines 1 and 12-13), but these referenced sections only state the existence of a deadlock problem and describe a detection/notification scheme. Applicants respectfully submit that none of these sections in Schoening teach, suggest or imply a technique to <u>avoid</u> deadlock, as recited by the present invention. Relative to Furlani, the final Office Action cited Col. 11, lines 40-42 thereof as teaching deadlock avoidance. Applicants submit that, compared to the deadlock avoided in present invention, the deadlock referenced in Furlani is of a different type. In Furlani, the deadlock discussion is limited to requiring a specific order of locking objects to avoid deadlock. In contrast, the present invention avoids deadlock by providing, not a specific order of locking objects, but an altered thread pool set to service the response (e.g., the altered thread pool set includes a second thread pool to process the response when the first thread pool has no threads available to process the response and the requests being processed on the first thread pool are waiting for the response to be fully processed). Thus, for the reasons stated above, applicants respectfully submit that the dependent claims presented herewith patentably distinguish over the applied art.

All claims are believed to be in condition for allowance, and such action is respectfully requested.

Should the Examiner wish to discuss this case further with applicants' attorney, the Examiner is invited to telephone their below-listed representative.

Respectfully submitted,

Kevin P. Radigan

Kevin P. Radigan
Attorney for Applicants
Registration No. 31,789

Dated: April 22, 2003

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579